Integrated Sensing and Communication Physical Layer Model Documentation

Steve Blandino, Tanguy Ropitault, Neeraj Varshney, Jian Wang, Anirud Sahoo, Camillo Gentile, Nada Golmie

WIRELESS NETWORKS DIVISION, COMMUNICATION TECHNOLOGY LAB

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY,

GAITHERSBURG, MD, USA

MARCH, 2022

Software Disclaimer

NIST-developed software is provided by NIST as a public service. You may use, copy and distribute copies of the software in any medium, provided that you keep intact this entire notice. You may improve, modify and create derivative works of the software or any portion of the software, and you may copy and distribute such modifications or works. Modified works should carry a notice stating that you changed the software and should note the date and nature of any such change. Please explicitly acknowledge the National Institute of Standards and Technology as the source of the software.

NIST-developed software is expressly provided "AS IS." NIST MAKES NO WARRANTY OF ANY KIND, EXPRESS, IMPLIED, IN FACT OR ARISING BY OPERATION OF LAW, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT AND DATA ACCURACY. NIST NEITHER REPRESENTS NOR WARRANTS THAT THE OPERATION OF THE SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT ANY DEFECTS WILL BE CORRECTED. NIST DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OF THE SOFTWARE OR THE RESULTS THEREOF, INCLUDING BUT NOT LIMITED TO THE CORRECTNESS, ACCURACY, RELIABILITY, OR USEFULNESS OF THE SOFTWARE.

You are solely responsible for determining the appropriateness of using and distributing the software and you assume all risks associated with its use, including but not limited to the risks and costs of program errors, compliance with applicable laws, damage to or loss of data, programs or equipment, and the unavailability or interruption of operation. This software is not intended to be used in any situation where a failure could cause risk of injury or damage to property. The software developed by NIST employees is not subject to copyright protection within the United States.

Contents

Abbreviations

AP Access Point.

BPSK Binary Phase Shift Key.

CFO carrier frequency offset.
CP Cyclic Prefix.

DCM Dual Carriere Modulation.
DDIR Double Directional Channel Impulse Response.
DFT Discrete Fourier Transform.

IDFT Inverse Discrete Fourier Transform.
ISAC Integrated Sensing and Communication.

LDPC Low Density Parity Check.

MCS Modulation and Coding Scheme.
mm-wave millimeter-wave.
MMSE Minimum Mean Square Error.
MSE Mean Squared Error.
MU-MIMO Multi-User Multiple-Input Multiple-Output.

NMSE Normalized Mean Squared Error.

OFDM orthogonal frequency-division multiplexing.

PAA Phased Array Antenna.
PHY Physical Layer.

Q-D Quasi-Deterministic.
QAM Quadrature Amplitude Modulation.
QPSK Quadrature Phase Shift Key.

SC single carrier.
STA Station.
SU-MIMO Single-User Multiple-Input Multiple-Output.

SU-SISO Single-User Single-Input Single-Output.
SVD Singular Value Decomposition.

ZF Zero Forcing.

Integrated Sensing and Communication Physical Layer Model Documentation

Integrated Sensing and Communication Physical Layer Model (ISAC-PLM) is an open-source implementation for link level simulation of millimeter-wave (mm-wave) wireless communication and sensing systems. ISAC-PLM models the Physical Layer (PHY) of IEEE 802.11ay, including a growing set of features, such as Multi-User Multiple-Input Multiple-Output (MU-MIMO) link level simulation, IEEE 802.11ay single carrier (SC)/orthogonal frequency-division multiplexing (OFDM) waveform generation, synchronization, channel estimation, carrier frequency offset (CFO) estimation and correction. Moreover, ISAC-PLM enables the simulation, design and test of future wireless systems integrating communication and sensing. Using the IEEE 802.11ay PHY, ISAC-PLM introduces micro-doppler processing to obtain estimation of range and velocity of moving targets.

1 Introduction

The ISAC-PLM is part of a collection of tools developed to support integrated sensing and communication system design and evaluation [1]. The ISAC-PLM models the end-to-end IEEE 802.11ay PHY processing based on a Matlab implementation[1] extending the communication functionalities with sensing features. Sensing through wireless communication networks aims at utilizing communication signals to detect and sense targets enabling application such as human presence detection, gesture recognition, or object tracking. The ISAC-PLM supports a native integration with the open-source NIST Q-D channel realization software to enable link-level simulations with realistic mm-wave channel models. The ray tracing method used in the NIST Q-D channel realization software, unlike conventional stochastic models, is 3-D environment and transceiver position specific, hence it enables an accurate end-to-end performance evaluation. The NIST Q-D channel realization software enables also the analysis of sensing

[1]Requirements: R2021b+, WLAN toolbox

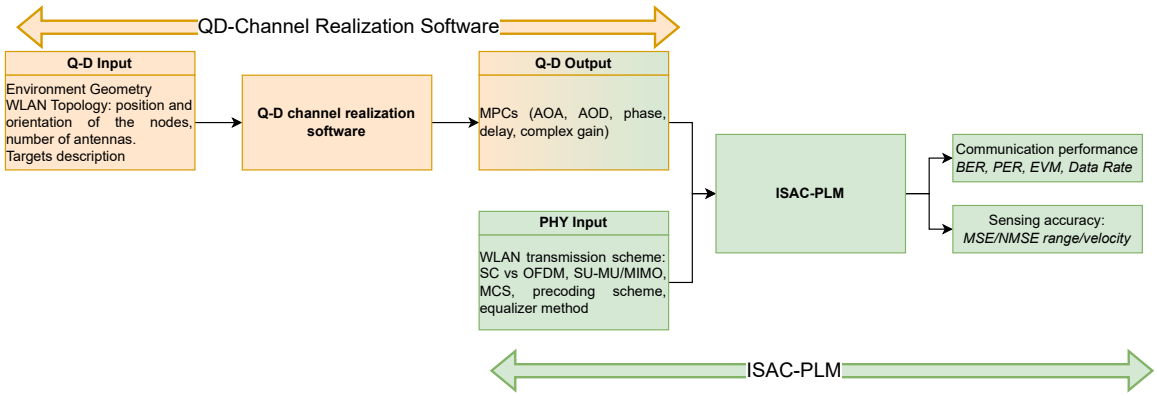

Figure 1: ISAC-PLM workflow.

applications, modeling the effect of moving targets on the channel.

Figure 1 presents an overview of the ISAC-PLM workflow and its integration with the NIST Q-D channel realization software.

The NIST Q-D channel realization software generates the mm-wave Double Directional Channel Impulse Response (DDIR) using the Quasi-Deterministic (Q-D) methodology [2], which requires the environmental definition to accurately perform the ray tracing. Hence, the geometry of the environment and the topology of the WLAN scenario, e.g., position of the nodes and targets are defined in the NIST Q-D channel realization software. The target model is given as input to the Q-D software, which returns the MPCs, both those generated by the interaction of the signal with the target (target related MPCs) and those generated by the interaction between the signal and the environment (target unreleated MPCs).

The channel model is used in the simulation of IEEE 802.11ay packets transmission and reception. The ISAC-PLM executes the digital baseband transmitter and receiver algorithms, and it can be configured with a wide range of parameters that define the properties of the communication system, the transceiver settings as well as the sensing processor settings. The communication receiver processor and the sensing processor return in output the performance of the communication link and the sensing accuracy.

The key communication features of the software can be summarized as follows:

- IEEE 802.11ay SC/OFDM waveform generation.
- IEEE 802.11ay spatial multiplexing schemes up to 8 streams: Single-User Single-Input Single-Output (SU-SISO), Single-User Multiple-Input Multiple-Output (SU-MIMO), MU-MIMO.
- Receiver signal processing algorithms: synchronization, channel estimation, carrier frequency offset (CFO) correction.
- Perform link-level bit error rate (BER), packet error rate (PER), data rate tests and analytical link-level spectral efficiency (SE).

The key sensing features of the software can be summarized as follows:

- Sensing signal processing algorithms: clutter removal, doppler processing, target detection, range and velocity estimation.
- Sensing accuracy analysis in terms of Mean Squared Error (MSE) and Normalized Mean Squared Error (NMSE).

2 Digital Baseband Transceiver and Sensing Processor

The digital transceiver extends the IEEE 802.11ad in the MATLAB WLAN toolbox[2] to be IEEE 802.11ay compliant. It supports SU-MIMO and-MU-MIMO for both OFDM and SC modes. It also supports several precoder and equalizer options, including Singular Value Decomposition (SVD), Zero Forcing (ZF) precoder, block diagonalized-ZF precoder, and Minimum Mean Square Error (MMSE) equalizer.

The EDMG frame is made of two main parts: the preamble and the data. The preamble containing known pilots sequence, is used to aid the receiver in correctly decoding the data. The proposed digital transceiver can perform synchronization of the received signal, as well as estimating the channel from the Golay pilot sequences provided in the IEEE 802.11ay packet preamble. Moreover, a sensing processor determines the presence of moving target using the evolution of the estimated channel.

2.1 Data Processing

Fig. 2 shows the baseband OFDM MU-MIMO signal processing of the EDMG digital transceiver.

As shown in Fig. 2(a), the blocks at the Access Point (AP) transmitter, including scrambler, Low Density Parity Check (LDPC) encoder, stream parser and constellation mapper, are operated on a per user basis; while the remaining processing in the transmit chains, including the spatial mapper, Inverse Discrete Fourier Transform (IDFT), and Cyclic Prefix (CP) insertion, are carried out per digital antenna (or digital chain). The digitally precoded MU-MIMO symbols of each transmit chain are OFDM modulated with a 512 point IDFT, followed by the CP insertion.

The Station (STA) receiver procedure is illustrated in Fig. 2(b). The OFDM symbols are demodulated by Discrete Fourier Transform (DFT) operation followed by the CP removal and a MMSE MIMO equalizer. Subsequently, the constellation demapping, stream deparsing and LDPC decoding are carried out to complete the PHY signal

[2]Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

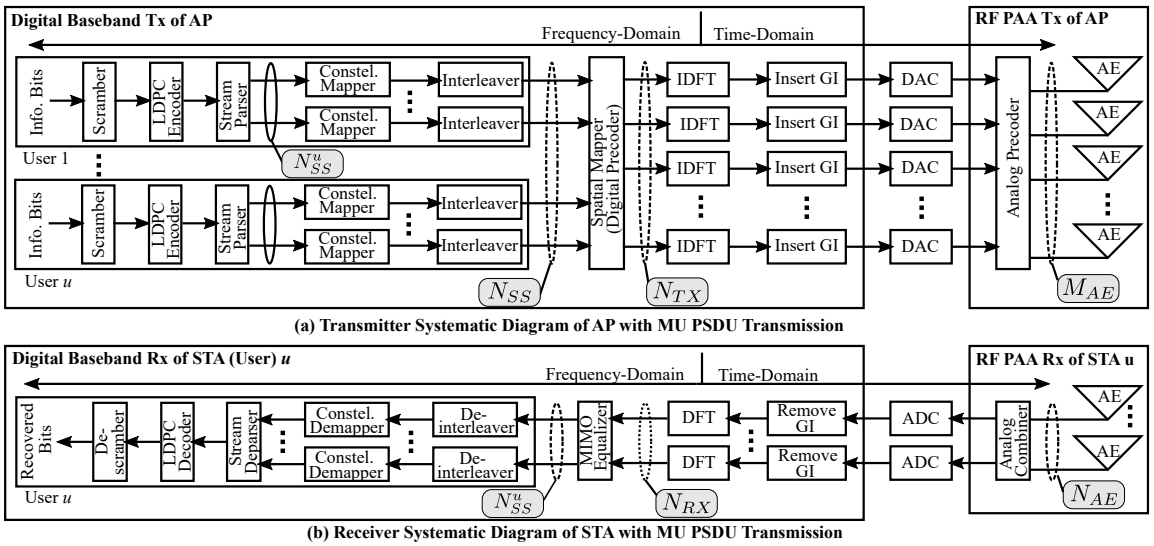

Figure 2: Transceiver diagram of IEEE 802.11ay OFDM mode, in MU-MIMO downlink transmission.

processing procedure. The LDPC encoder and decoder as well as constellation mapper and demapper support a variety of Modulation and Coding Schemes (MCSs) for OFDM mode transmission with different coding rates to comply with the IEEE 802.11ay standard, including Dual Carriere Modulation (DCM)-Binary Phase Shift Key (BPSK), DCM-Quadrature Phase Shift Key (QPSK), 16-Quadrature Amplitude Modulation (QAM), and 64-QAM.

2.2 Preamble Processing

The preamble contains known pilots sequence, namely Golay sequences, in the legacy short time field (L-STF), in the EDMG-STF, in the legacy channel estimation field (L-CEF) and in the EDMG-CEF. STF and CEF are used in the communication receiver signal processing, for time and frequency synchronizations and channel state information (CSI). These operations are achieved in multiple steps:

- The frame is detected and synchronized by finding the peak of the correlation between the received signal and the known STF pilots.
- Coarse frequency offset is computed by comparing the phase changes into the peaks of the STF correlation.
- Joint frequency offset and channel estimation is performed by correlating the received CEF with the known CEF.

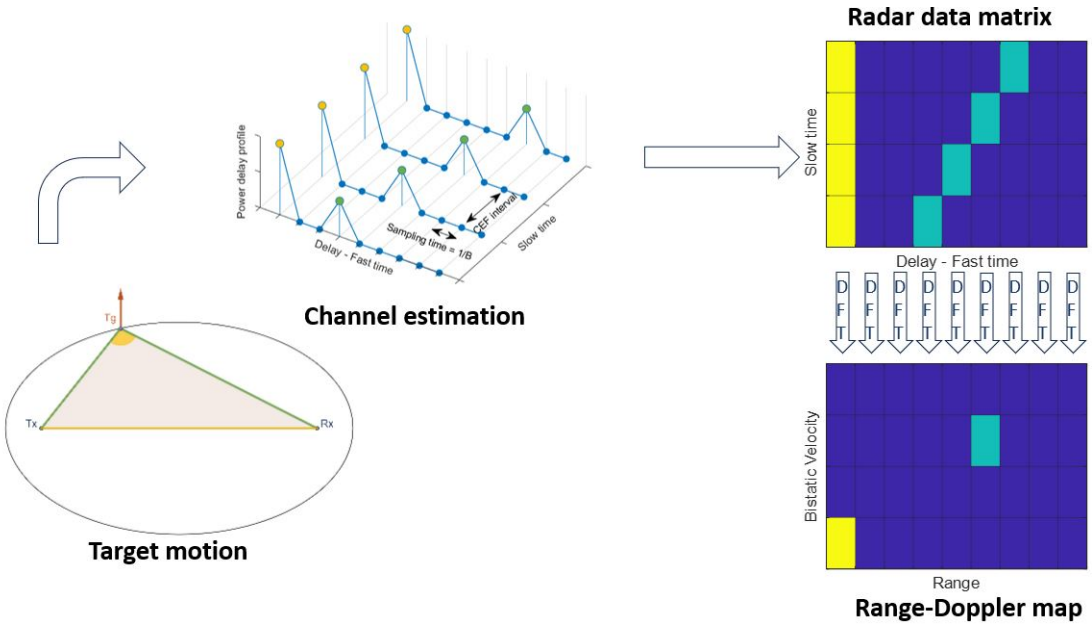

Figure 3: Doppler processing using channel estimation.

2.3 Sensing Processing

The sensing processor is available for SISO communication links. ISAC-PLM considers a CSI based sensing method. The CSI, which is already available in the conventional IEEE 802.11ay system, is used to sense the environment by tracking the channel variations over time. Assuming static transmitter and receiver, a perturbation of the CSI can only be caused by a change in the environment, such as a moving target. The sensing initiator is the sensing receiver, i.e, it will receive the PPDU sent by the sensing responder, which is the sensing transmitter. The sensing initiator is also the sensing processor, beyond a communication receiver.

From a remote sensing perspective, the channel estimated using the preamble can be seen as echoes from the targets and the environment. The delay of the echoes from the targets MPCs are proportional to the bi-static distance, i.e. the distance the signal travels from the transmitter to the receiver scattering out of the target.

As the IEEE 802.11ay packets are sent continuously over time, the sensing processor can build the radar data matrix, collecting the estimated CSI at each packet reception in a 2-dimensional matrix, i.e., the delay (referred also as fast time) and the evolution over the time (referred also as also slow time). To obtain the velocity of the targets, a discrete Fourier transform (DFT) is applied on the slow time dimension of the radar data matrix to allow the estimation of the target speed. A visual representation of the Doppler processing is shown in Figure 3.

Null Doppler values in the range-Doppler map, i.e. null velocity, are considered static clutter, i.e., echoes coming from the static environments, thus they are not rele-

vant to the remote monitoring. They can be filtered out, by removing the continuous component along the slow time dimension of the radar data matrix.

3 Software Structure

3.1 Block Diagram of ISAC-PLM

The ISAC-PLM is organized in three main building blocks as shown in Figure 4:

- The EDMG transmitter is in charge to encode randomly generated bits by LDPC, modulate encoded bits to a constellation and apply SC or OFDM modulation to obtain the waveform.
- The EDMG receiver gets in input the waveform passed through the channel model with added white noise. The receiver performs frame synchronization, SC/OFMD demodulation and channel estimation. The channel estimation is used to obtain the equalizer to retreive the constellation. Demodulation and LDPC decoding allows to retreive the information bits.
- The sensing processor uses the channel estimation already computed to perform sensing. The channel estimation is first filtered trough a clutter removal filter, that eliminates the static background. After that the doppler processing enables velocity and range estimation.

3.2 Folder and Code Structure

3.2.1 Directory Structure

The directory structure is shown in Figure 5. The MATLAB code is located in the folder `isac-plm` containing the main script `main.m` to run the software as a communication link and the the main script `mainIsac.m` to run the software as Integrated Sensing and Communication (ISAC) link. The folder `example` collects predefined configurations. Each of the examples has an `Input` folder defining the configuration files and an `Output` folder that is generated by the ISAC-PLM with the result of the simulation. The user can define other folders with custom simulation settings.

The folder `isac-plm\src` contains the Matlab code and includes several sub-folders:

- `channel`: includes the function to generate different channel models, as well as the function to convert the MPCs obtained from the NIST Q-D channel realization software to a tapped delay line (TDL) channel model.
- `config`: configuration functions. It includes the file to load the input configuration and set default values.

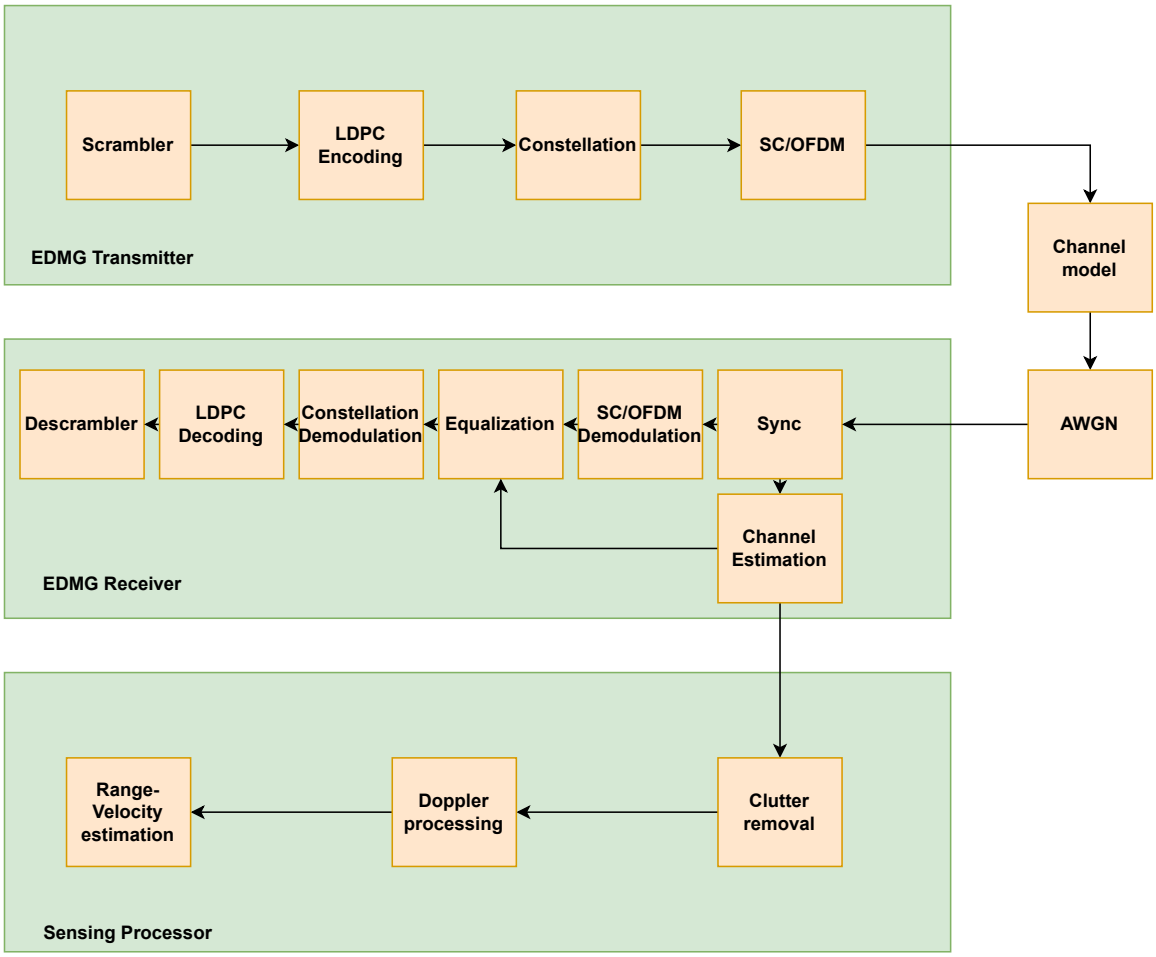

Figure 4: ISAC-PLM block diagram.

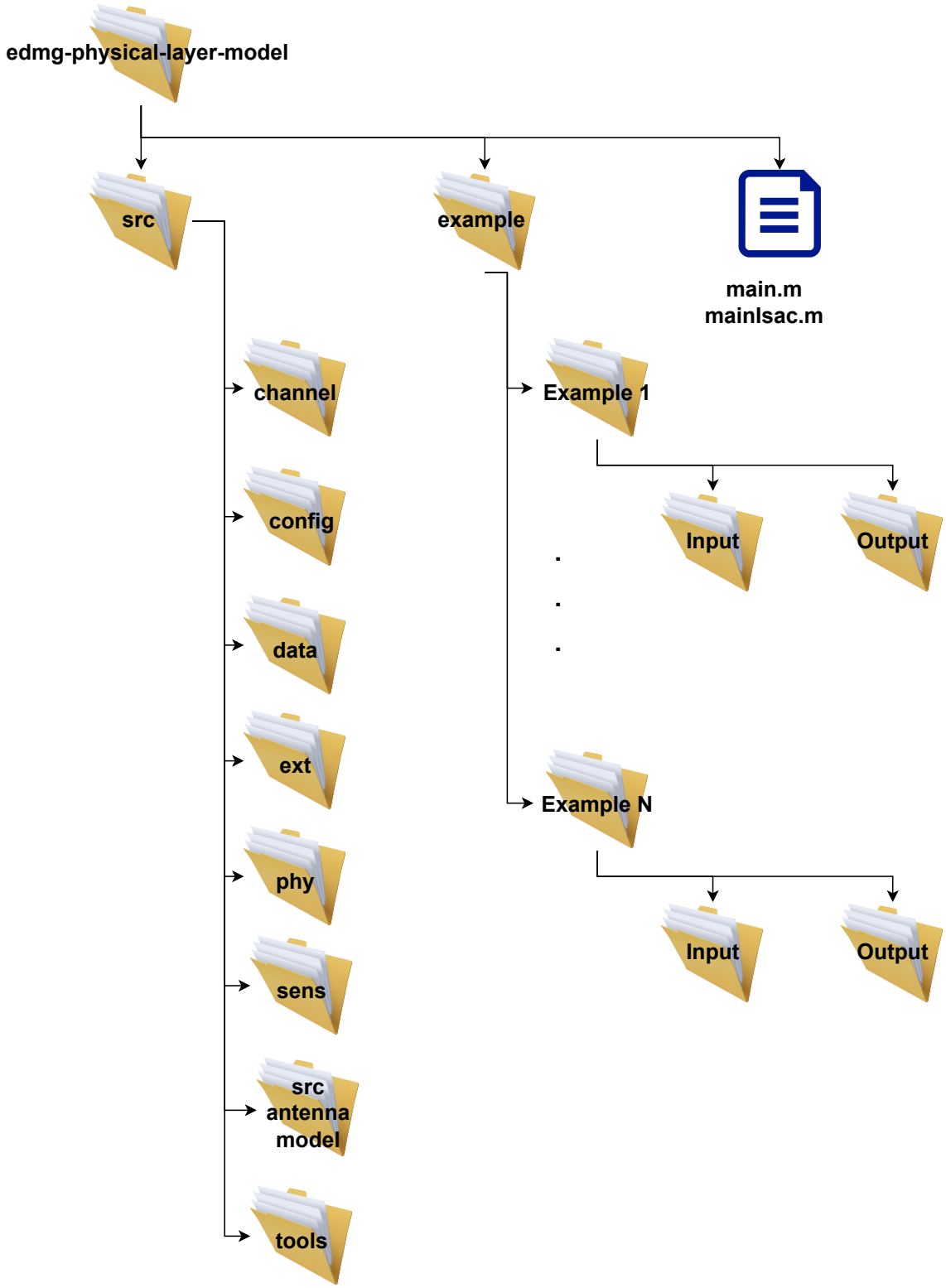

Figure 5: ISAC-PLM folder structure.

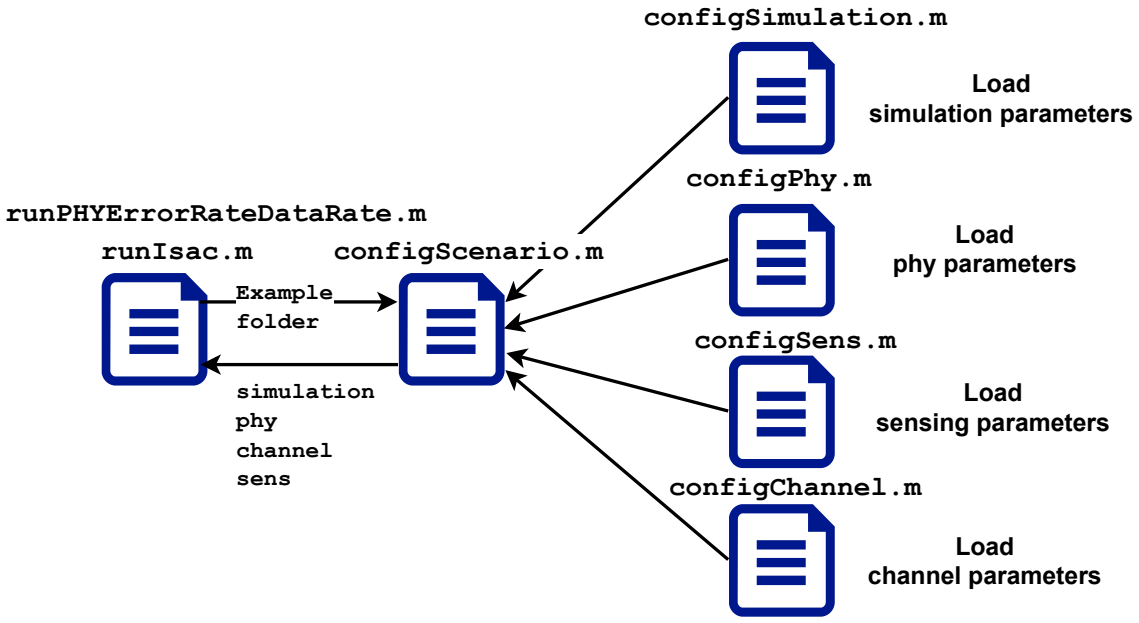

Figure 6: Code structure.

- `data`: stores data needed to run the software, e.g., angle codebook.
- `ext`: external code files.
- phy: link-level simulations functions.
  - +nist: MathWorks authored codes revised by NIST.
- `sens`: collects the sensing functions.
- `tools`: collects helper functions.
- `src-antenna-model`: collects the functions to read the angle codebook.

3.2.2 Code Structure

The script main.m or mainIsac.m are used to launch the simulation. main.m invokes runPHYErrorRateDataRate.m and runPHYSpectralEfficiency.m to runs the link-level simulation to obtain BER, PER, data rate and spectral efficiency. mainIsac invokes runIsac to obtain, beyond BER, PER, data rate, the sensing accuracy in case of moving targets.

The ISAC-PLM uses four major structures throughout the implementation. These structures are:

- `simulation`: it contains information about the simulation and the simulator setting. It includes parameter to set noise value, non-idealities, or to enable debug mode or plots.
- `phy`: the structure contains all the information relative to the transceiver operation and the algorithms used in the PHY.
- `sens`: this structure is used when using the integrated sensing and it contains the setting of the sensing processor.
- `channel`: it includes parameters related to the channel model.

These structures are constructed with dedicated configuration files. The configuration files provided as input and described in Section 4.1 are loaded using the functions `configSimulation.m`, `configPhy.m`, `configSens.m` and `configChannel.m`. In case some input parameter is missing in the configuration, these function initializes the parameter to a default value. The code structure is illustrated in Figure 6: the function `runPHYErrorRateDataRate` or the function `runIsac` invokes the `configScenario.m` function passing the example folder where the input files are defined. `configScenario.m` returns in output the structures used throughout the software.

After loading the configuration the ISAC-PLM uses three main building blocks, described in Figure 4, implemented in three different functions:

- The function `edmgTx.m` generates the IEEE 802.11ay transmit waveform.
- The function `edmgRx.m` implements the receiver processing.
- The function `sensingProcessing.m` implements the sensing processing on the estimated channel state information at the receiver.

4 Usage description

This section describes how to setup the ISAC-PLM and how to analyze the results obtained.

4.1 Input

To execute a simulation scenario in the ISAC-PLM , a scenario folder must be defined in `isac-plm\examples`. The scenario folder must contain an input folder with the configuration of the simulation. The input folder must be properly defined in the folder `isac-plm\examples\exampleFolder\Input`.

The configuration is broken down into four configuration files relative to the main structures used in the simulator: simulation, PHY, channel and sensing. Each structure

is defined by a `.txt` file containing in each row a pair parameter-value separated by a tab space. While the input parameters of each file are described in the following, an overview of the accepted parameters is provided in appendix A.

4.1.1 Simulation Configuration

The simulation configuration aims at selecting the properties of the simulation that the users want to analyze and chose the operation mode of the simulator.

The file `configSimulation.txt` may contain the following fields:

- `debugFlag`: define the software operating mode.

  0 for simulation,1 for debugging. If the field is not defined debugFlag is set to 0.

- `pktFormatFlag`:select the packet format.

  0 for PPDU format (header + data),1 for PSDU format (data-field only). Using the PPDU format the receiver does not use perfect channel knowledge to operate but it estimates the channel processing the header. If the field is not defined pktFormatFlag is set to 1 and the channel is perfectly known at both transmitter and receiver.

- `dopplerFlag`: doppler flag.

  0: Doppler off (block fading). 1: Doppler ON. (Default = 0)

- `snrMode`: SNR definition.

  Ratio of symbol energy to noise power spectral density EsNo, Ratio of bit energy to noise power spectral density EbNo, Signal-to-noise ratio (SNR) per sample SNR. (Default value: EsNo)

- `snrAntNormFlag`: SNR Rx antenna normalization flag.

  The receive signal is normalized such that the receive power at each antenna is 1 in average. 1: The receive signal is normalized such that the receive power at each antenna is 1. (Default 0)

- `snrRange`: Simulation SNR range in dB.

  Define the SNR range in dB specified as a 1-by-2 vector of scalar. (Default [0 20])

- `snrStep`: Define the SNR step in dB specified as a positive scalar.

  The simulation SNR points are generated in the range snrRange with a step of snrStep, i.e. snrRange(1):snrStep:snrRange(2). (Default 1)

- `maxNumErrors`:Maximum number of error specified as a positive integer. (Default = 100)

- `maxNumPackets`: Maximum number of packets specified as positive integer. (Default = 1000)
- (ISAC) `nTimeSamp`: Specify the length of the simulation in time. This parameter must be lower or equal to the number of channel samples in time provided as input (see Channel Configuration).
- (ISAC) `plotRangeDopplerMap`: plot range Doppler map for each SNR value if `plotRangeDopplerMap` = 1 and automatically save them in `.\ example\-exampleFolder\Output` folder. It this field is not defined, `plotRangeDopplerMap` is set as 1.
- (ISAC) `plotVelocity`: generate micro-Doppler vs time plot for each SNR value if `plotVelocity` = 1 and automatically save them in `.\example\exampleFolder\Output` folder. It this field is not defined, `plotVelocity` is set as 1.
- (ISAC) `plotRange`: generate range vs time plot for each SNR value if `plotRange` = 1 and automatically save them in `.\example\exampleFolder\Output` folder. It this field is not defined, `plotRange` is set as 1.

4.1.2 PHY Configuration

The PHY configuration defines the properties of the PHY and the algorithms the PHY uses in simulation.

The file `phyConfig.txt` may contain the following fields:

- `phyMode`: PHY mode.

  `phyMode` is specified as OFDM or SC. (Default OFDM)
- `lenPsduByt`: Length of PSDU.

  `lenPsduByt` is in byte specifies as positive scalar. (Default = 4096)
- `giType`: Guard interval length.

  `giType` is specified as Short, Normal or Long.
- `numSTSVec`: Number of spatial-time streams.

  `numSTSVec` specified as 1-by-STAs vector of positive integers in the range 1-8 such that sum(`numSTSVec`)<=8. (Default = 1)
- `smTypeNDP`: Spatial mapping type for non-data packet (preamble only, non data-field of PPSU)

  Specified as Hadamard, Fourier, Custom or Direct. (Default is Direct)

- `smTypeDP`: Spatial mapping type for data packet (PSDU, data-field of PPSU).

  smTypeDP specified as Hadamard, Fourier, Custom or Direct. (Default is Direct)

- `mcs`: Modulation and coding scheme (MCS) Index.

  mcs is specified as index in the range 1-20 (21 for SC). (Default = 6).

- `analogBeamforming`: Analog beamforming scheme.

  `analogBeamforming` specified as `maxAllUserCapacity`, `maxMinAllUserSV`, `maxMinPerUserCapacity`, `maxMinMinPerUserSV`, respectively. (Default is `maxAllUserCapacity`) This value is used if chanModel in simulationConfig.txt is specified as NIST. Application of analog beamforming is performed with an external MATLAB application, not included in this release.

- `dynamicBeamNumber`: Dynamic stream allocation.

  `dynamicBeamNumber` selects only the streams among `numSTSVec` with high SINR. `dynamicBeamNumber` is specified as a scalar between 0-20 indicating the condition number of a SU-MIMO matrix. This value is used if `chanModel` in simulationConfig.txt is specified as NIST. (0: OFF) (Default 0).Dynamic stream allocation is based on the analog channel state information and it is performed with an external MATLAB application, not included in this release.

- `processFlag`: transceiver digital processing flag.

  `processFlag` selects specific MIMO signal processing for OFDM/SC mode transmitter and receiver. processFlag is specified as a scalar between 0-5 (Default 0).

  - For the transmitter processing,
    - processFlag = 0, supports both OFDM and SC SISO and SU-MIMO without transmitter precoding.
    - processFlag = 1, OFDM supports SU- and MU-MIMO with frequency-domain precoding based on regularized zero-forcing (RZF) criteria; SC supports SU-MIMO with time-domain one-tap precoding based on RZF criteria.
    - processFlag = 2, OFDM supports SU-MIMO with frequency-domain precoding based on based on SVD and RZF filtering; SC supports SU-MIMO with time-domain one-tap precoding based on SVD and RZF filtering.
    - processFlag = 3, OFDM supports MU-MIMO with frequency-domain precoding based on based on SVD and RZF filtering; SC supports SU-MIMO with time-domain one-tap precoding based on SVD and RZF filtering.

        * processFlag = 4, OFDM supports SU- and MU-MIMO with frequency-domain precoding based on based on block diagonalization and ZF filtering (BD=ZF); SC supports SU-MIMO with time-domain one-tap precoding based on BD-ZF.
        * processFlag = 5, SC supports MU-MIMO with time-domain multi-tap precoding with ZF.
    - For the receiver processing, all processFlag = 0 5 support OFDM/SC joint frequency-domain equalization and MIMO detection based on linear minimum mean squared error (MMSE) criteria.

- `symbOffset`: Symbol sampling offset.

  `symbOffset` is specified as values from 0 to 1. When `symbOffset` is 0, no offset is applied. When `symbOffset` is 1 an offset equal to the GI length is applied. (Default 0.75)

- `softCsiFlag`: Demodulation with soft channel state information flag.

  `softCsiFlag` is specified as 0 for inactivated or 1 for activated. (Default 1 for OFDM and 0 for SC)

- `ldpcDecMethod`: LDPC decoding method.

  `ldpcDecMethod` is specified as `norm-min-sum` or `bp`, respectively. (Default norm-min-sum)

4.1.3 Sensing Configuration

The sensing configuration aims at defining the parameters used in the sensing processor. The file `sensConfig.txt` may contain the following field:

- `pri`: pulse repetition interval, specifies the interval between each packet as a positive real. Default 0.0005.
- `dopplerFftLen`: specifies the doppler FFT length as a positive integer. Default 64.
- `pulsesCpi`: pulses in a coherent processing interval, specifies the number of packet to collect before to compute the doppler FFT. Default 64.
- `window`: FFT window specifies as `rect`, `hamming`, `blackmanharris`, `gaussian`. Default `rect`.
- `windowLen`: Split data before doppler FFT in overlapping windows of length `windowLen`. Default 64.
- `overlap`: Specifies the percentage of the window overlapping between 0 and 1. Default 0.

4.1.4 Channel Configuration

The channel configuration specifies the properties of the channel over which the packets are transmitted.

- To evaluate the PHY performance, several channel models can be used, such as AWGN, Rayleigh or Q-D models.
- To evaluate the performance of an ISAC system, the NIST Q-D channel realization model must be used, since it provides a deterministic description targets.

The channel model is configured with the file `channelConfig.txt`. The file `channelConfig.txt` must contain the field `chanModel` specified as `AWGN, Rayleigh, NIST, sensNIST`. The labels `NIST` and `sensNIST` refers to channels obtained using the Q-D channel realization software without and with moving targets respectively. To use `NIST` please contact the authors for additional dataset. As the configuration of each channel model might have different parameters the file `channelConfig.txt` contains different fields according with the channel model selected.

Rayleigh Channel Parameter Configuration

The file `channelConfig.txt` may contain the following fields:

- `numTaps`: Number of channel taps.

  Number of channel taps specified as a positive integer. If the field is not defined `numTaps` is set to 10.

- `pdpMethodStr`: Power delay profile for tapped-delay line (TDL) channel model.

  `pdpMethodStr` is specified as `PS`, `Equ` or `Exp`, using phase shift, equal power or exponential power, respectively.

- `tdlTypeChannel` Interpolation Method of CIR.

  `tdlTypeChannel` is specified as `Impulse` or `Sinc`.(Default `Impulse`)

NIST Q-D Channel Parameter Configuration

The ISAC-PLM provides a dataset of system level channels obtained using the NIST-QD channel realization software and provided upon requests. After ray tracing, the MPCs are filtered trough a PAA antenna model and analog beamforming algorithms. The system level channels can be provided for different indoor and outdoor environments as well as for different PAA models and analog beamforming schemes. Figure 7 shows a living room environment and the spatial filtering on the MPCs provided by the analog beamforming.

The file `channelNistConfig.txt` loading one of the predefined channel, may contain the following fields that allows to load the correct dataset:

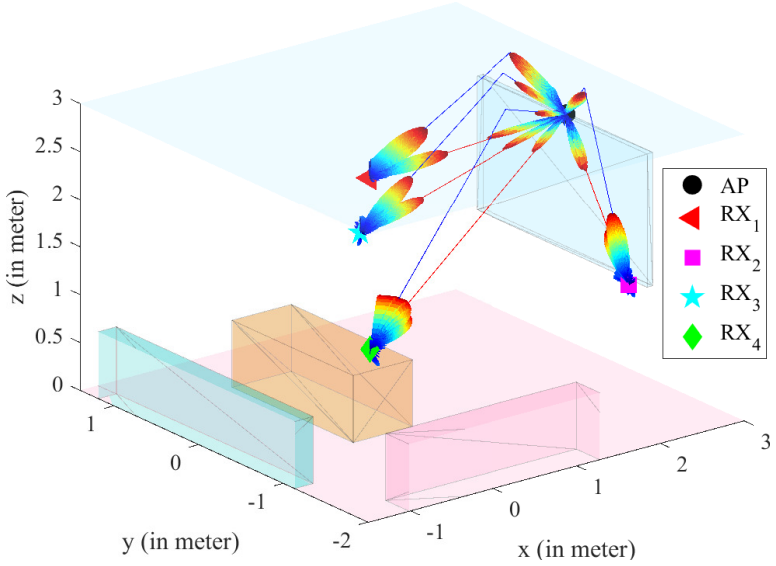

Figure 7: Indoor Q-D channel model for a living room scenario and spatial filtering provided by analog beam-steering of Phased Array Antenna (PAA).

- `environmentFileNamechannel` environmentFileNamechannel is specified as `LR` for lecture room, `OAH` for open area hotspot or `SC` for street canyon. If the field is not defined environmentFileName is set to `LR`.
- `totalNumberOfReflections`: Reflection order. totalNumberOfReflections is specified as a positive integer. If the field is not defined totalNumberOfReflections is set to 2.
- `tdlTypeChannel` Interpolation Method OF TDL CIR filtering. `tdlTypeChannel` is specified as Impulse or Sinc.

Moreover, the antenna model can be configured for each node in the system thanks to the file `paaConfigNodeX.txt` where `X` represents the node index from 0. `paaConfigNodeX.txt` may contain the following fields:

- `numAntenna`: Total number of antenna element in the node.

  `numAntenna` is specified as a positive integer. If the field is not defined `numAntenna` is set to 16.
- `Geometry`: Geometry of the antenna array.

  `Geometry` is specified as `UniformLinear` or `UniformRectangular`.
- `numAntennaVert`: Total number of antenna element in the vertical direction.

  `numAntennaVert` is specified as a positive integer. If the field is not defined `numAntennaVert` is set to 4.

NIST QD Channel for sensing

ISAC-PLM provides a native integration with the Q-D channel realization software when using the label `sensNIST`. The file `channelConfig.txt` must contain only the mandatory field `chanModel` specified as `sensNIST`. A subfolder `qdChannel` must be created. This folder contains the output of the Q-D channel realization software, i.e. the files `qdOutput.json` and `qdTargetOutput.json`. An example on how to use the ISAC-PLM together with the Q-D channel realization software is provided in Section 5.1.

AWGN

To use an AWGN channel it is enough to specify the property `chanModel` as `AWGN` in the configuration file `channelConfig.txt`.

4.2 Output

ISAC-PLM provides as output the results of the link-layer simulation. The evaluation of the communication system is provided in terms of BER, PER, EVM and data rate, while an ISAC system is evaluated also in terms of the sensing accuracy, hence, velocity and range MSE and NMSE are provided.

The results of the simulation are stored in `isac-plm\examples\exampleFolder\Output`. The file `isac-plm-ws.mat` stores the following information:

- `berIndiUser`: BER for each individual user.
- `perIndiUser`: PER for each individual user.
- `evmIndiUser`: EVM expressed in dB for each individual user.
- `berAvgUser`: Average BER among users.
- `perAvgUser`: Average PER among users.
- `evmAvgUser`: Average EVM among users.
- `gbitRateIndiUser`: Data rate for each individual user.
- `gbitRateAvgUser`: Average data rate for each individual user.
- `gbitRateSumUser`: Sum data rate.
- (ISAC) `sensing`
  - `rangeNMSE`,`velocityNMSE`: range and velocity NMSE
  - `rangeMSE`,`velocityMSE`: range and velocity MSE

  - `rEst`,`vEst`: range and velocity estimation.
  - `doppler`: range doppler map.

In case of an ISAC simulation ISAC-PLM provides in `\Output\Sensing` the results in JSON format. Two files are provided:

- `sensingInfo.json`: stores variable relative to time and velocity scale as well as ground truth information of range and velocity.
  - `velocityGrid`: vector of velocities bins expressed in m/sß.
  - `fastTimeGrid`: vector of delay bins relative to sync point expressed in s.
  - `delayShift`: delay shift expressed in s to map the relative fast time `fastTimeGrid` into an absolute temporal scale `fastTimeGridAbs` as `fastTimeGridAbs` = `fastTimeGrid` + `delayShift`.
  - `slowTimeFftGrid`: vector of slow time bins at which doppler fft is computed expressed in s.
  - `slowTimeGrid`: vector of slow time bins expressed in s.
  - `gtRange`: ground truth range expressed in m.
  - `gtVelocity`: ground truth velocity expressed in m/s.
- `sensingResults.json`: stores the sensing accuracy, estimated range and doppler and the raw range doppler map:
  - `snr`: SNR
  - `nmseRange`: NMSE range expressed in dB scale.
  - `nmseVelocity`: NMSE velocity expressed in dB scale.
  - `mseRange`: MSE range expressed in dB scale.
  - `mseVelocity`: MSE velocity expressed in dB scale.
  - `rangeEstimate`: range estimation expressed in m.
  - `velocityEstimate`: velocity estimation expressed in m/s.
  - `rangeDopplerMap`: raw range doppler map.

Moreover, for a quick analysis of the sensing results, in case of an ISAC simulation, ISAC-PLM provides three output plots in `\Output\Sensing\Figures`:

- Micro-doppler: evolution of the velocity over time.
- Range doppler map
- Range time: evolution of the range over time.

5 Examples

The following example demonstrate how to use the tools provided in [1] to simulate an ISAC system.

To follow these examples simply checkout both Q-D channel realization software and ISAC-PLM in `yourfolder\`.

The folder structure will be as follow:

- `yourfolder\qd-realization`
- `yourfolder\isac-plm`

5.1 Point target

This example starts with the configuration of the NIST QD channel realization software. A single dimensionless point target moving is ray traced between one transmitter and one receiver in a 3-D empty room. The ray traced multipath components provide a deterministic channel model, which is specific for the trasmitter and receiver location, the environment and the motion of the target analyzed. The channel model is used as input configuration of ISAC-PLM to simulate a SISO-SC ISAC link.

The input configuration and the output results are already provided, however in the following, a step-to-step tutorial is provided to re-design the example by using Matlab. The configuration of the software can be obtained also with other tools, since the input parameters are stored in text files.

5.1.1 Deterministic Channel Model

The Q-D channel configuration folder of the Point Target example is already provided in `qd-realization\src\examples\PointTarget`.

To re-create the Point Target example, generate a new input folder in which the configuration files will be added:

```
exampleFoler = '<>\qd-realization\src\examples\NewPointTarget';
inputFolder = ([exampleFolder , '\Input']);
mkdir(inputFolder)
```

3-D Scenario Configuration

- Specify the indoor 3-D map in XML format.

  In the example folder `qd-realization\src\examples\PointTarget\Input` an empty living room of dimension $7 \times 7 \times 3\,\mathrm{m}$ is provided in `LivingRoom.xml`, however a custom map can be created. For a tutorial on how to create a custom map, please refer to [3], otherwise just paste the provided file.

- Specify the nodes position in `NodePositionX.dat`

  The SU-SISO link is composed by two nodes, one transmitter and one receiver. The transmitter (node 0) is placed at (-3.9,0,2.8) m while the receiver (node 1) is placed at (3,0,2.8) m.

  The configuration files can be written as:

```
writematrix([-3.9,0,2.8],[inputFolder,\'NodePosition0.dat'])
writematrix([3,0,2.8],[inputFolder,\'NodePosition1.dat'])
```

- Specify the point target trajectory in `TargetBase0.dat`

  The target is moving over time, hence a list of 3-D coordinates need to be provided. In this example a linear motion is used and it can be generated as:

```
x = linspace(-1, 0, 512)';
y = linspace(-1, 0, 512)';
z = linspace(1, 1, 512)';
writematrix([x,y,z], [inputFolder,'\TargetBase0.dat'])
```

- Specify the parameter configuration of the Q-D channel realization software in `paraCfgCurrent.txt`.

  The Q-D is configured to generate 256 channel realization in the indoor living room map provided. The channel is configured to have a single reflection order from the environment and the target rays are not interacting with the environment (no target ghost on the reflecting surfaces of the environment). The total motion is assumed to be 2 seconds. The integration with the ISAC-PLM is enabled by writing the QD channel realization software in JSON format and by returning the target rays.

```
%Environment file
cfg.ParameterName{1,1} = 'environmentFileName';
cfg.ParameterValue{1,1} = 'LivingRoom.xml';
%Number of channel realization to generate
cfg.ParameterName{2,1} = 'numberOfTimeDivisions';
cfg.ParameterValue{2,1} = 256;
%Environment reflection order
cfg.ParameterName{3,1} = 'totalNumberOfReflections';
cfg.ParameterValue{3,1} = 1;
```

```
%Target reflection order
cfg.ParameterName{4,1} = 'totalNumberOfReflectionsSens';
cfg.ParameterValue{4,1} = 0;
%Simulation time in seconds
cfg.ParameterName{5,1} = 'totalTimeDuration';
cfg.ParameterValue{5,1} = 2;
%Output file format
cfg.ParameterName{6,1} = 'outputFormat';
cfg.ParameterValue{6,1} = 'json';
%Store a dedicated target MPC file
cfg.ParameterName{7,1} = 'writeTRayOutput';
cfg.ParameterValue{7,1} = 1;
%Write phase of MPCs
cfg.ParameterName{8,1} = 'enablePhaseOutput';
cfg.ParameterValue{8,1} = 1;
%Write Configuration File
writetable(struct2table(cfg), ...
[inputFolder, '\paraCfgCurrent'], "FileType", 'text',...
'delimiter', '\t')
```

Ray tracing

To perform the ray tracing:

- Open `main.m` in `yourfolder\qd-realization\src`
- Set `scenarioNameStr = 'examples\NewPointTarget'`
- Run `main.m`

5.1.2 ISAC system and signal processing

The ISAC-PLM configuration example is provided in `\isac-plm\examples\PointTarget`.

To re-create the Point Target example, generate a new input folder in which the configuration files will be added:

```
plmExampleFolder = '\isac-plm\examples\NewPointTarget';
inputFolder = ([plmExampleFolder , '\Input']);
mkdir(inputFolder)
```

ISAC configuration

- Specify the PHY configuration in `phyConfig.txt`

  Configure a SC SU-SISO comminication link that uses a 16-QAM.

```
% Set SC mode
cfgPhy.ParameterName{1,1} = 'phyMode';
cfgPhy.ParameterValue{1,1} = 'SC';
% Set SISO
cfgPhy.ParameterName{2,1} = 'numSTSVec';
cfgPhy.ParameterValue{2,1} = 1;
% Set MCS
cfgPhy.ParameterName{3,1} = 'mcs';
cfgPhy.ParameterValue{3,1} = 12;
% Write Configuration File
writetable(struct2table(cfgPhy), ...
[inputFolder, '\phyConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Specify the sensing configuration in `sensConfig.txt`.

  Configure the sensing processor to use range-doppler processing using an FFT length of 64 and for a CPI of 64. No STF overlap is used in this example and no data windowing.

```
% Set pulse repetition interval
cfgSens.ParameterName{1,1} = 'pri';
cfgSens.ParameterValue{1,1} = 0.0005;
% Set FFT-lenght for doppler processing
cfgSens.ParameterName{2,1} = 'pri';
cfgSens.ParameterValue{2,1} = 256;
% Set FFT-window for doppler processing
cfgSens.ParameterName{3,1} = 'pri';
cfgSens.ParameterValue{3,1} = 'rect';
% Set FFT-window lenght
cfgSens.ParameterName{4,1} = 'windowLen';
cfgSens.ParameterValue{4,1} = 64;
% Set FFT-window overlap
cfgSens.ParameterName{5,1} = 'windowOverlap';
cfgSens.ParameterValue{5,1} = 0;
```

```
% Set CPI
cfgSens.ParameterName{6,1} = 'pulsesCpi';
cfgSens.ParameterValue{6,1} = 64;
```

- Specify the simulation setting in `simulationConfig.txt`

  Set the simulator to transmit 256 complete packets (data+preamble) with an SNR of 20 dB.

```
% Use preamble in packet transmission
cfgSim.ParameterName{1,1} = 'pktFormatFlag';
cfgSim.ParameterValue{1,1} = 0;
% Define length of the simulation
cfgSim.ParameterName{2,1} = 'nTimeSamp';
cfgSim.ParameterValue{2,1} = 256;
% Define SNR
cfgSim.ParameterName{3,1} = 'snrRange';
cfgSim.ParameterValue{3,1} = 20;
% Write Configuration File
writetable(struct2table(cfgSim), ...
[inputFolder, '\simulationConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Channel configuration in `channelConfig.txt`

  Configure the ISAC-PLM to read the output of the Q-D channel realization software:

```
cfgChan.ParameterName{1,1} = 'chanModel';
cfgChan.ParameterValue{1,1} = 'sensNIST';
% Write Configuration File
writetable(struct2table(cfgChan), ...
[inputFolder, '\channelConfig'], "FileType", 'text',...
'delimiter', '\t')
```

- Copy Q-D output to ISAC-PLM input

  Copy the Q-D output (`qdOutput.json` and `qdTargetOutout.json`) to the ISAC-PLM input folder `isac-plm\examples\NewPointTarget\Input\qdChannel`.

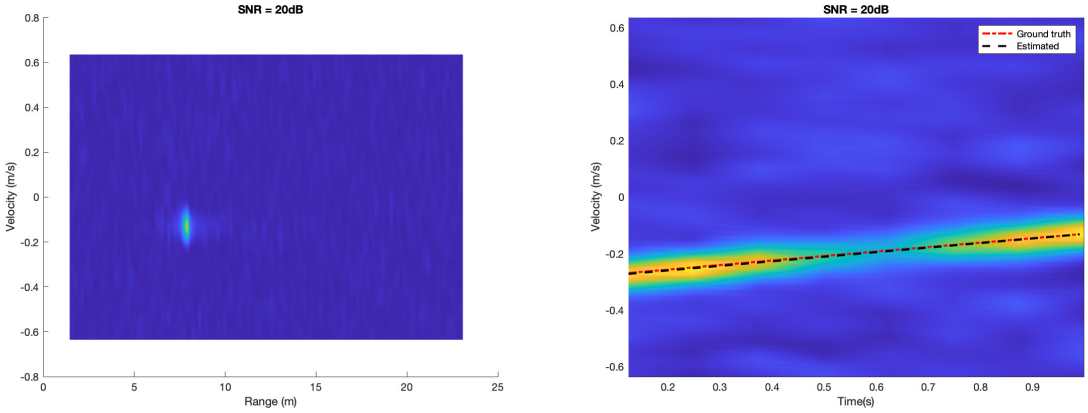

Figure 8: Point target sensing results.

ISAC simulation

To perform the isac simulation:

- Open `mainIsac.m` in `yourfolder\isac-plm\src`
- Set `scenarioNameStr = 'examples\NewPointTarget'`
- Run `mainIsac.m`

ISAC results

Beyond the output files stored in `\Output`, several figures are available in `\Output\-Sensing\Figures` for a quick interpretation of the sensing results. Figure 8 shows the range doppler map and the micro-doppler. The range doppler map on the left, shows a single peak that correspond to the velocity and range of the target. The micro-doppler on the right shows the variation of the detected velocity over time.

Appendix A

List of Input Parameters

Table A.1: Tunable properties of `simulation` in `simulationConfig.txt`.

Parameter	Description	Data Type	Valid Range	Default
`debugFlag`	Enables debug mode	double	$\{0, 1\}$	0
`pktFormatFlag`	Packet type	double	$\{0, 1\}$	1, 0*
`dopplerFlag`	Enables Doppler effect	double	$\{0, 1\}$	0
`snrMode`	SNR definition	char — string	'EsNo', 'EbNo'	'EsNo'
`snrAntNormFlag`	SNR normalization	double	$\{0, 1\}$	0
`snrRange`	SNR range	1-by-2 double	Positive Values	[0 20]
`snrStep`	SNR step	double	Positive integer	1
`maxNumErrors`	Max number of errors	double	Positive integer	100
`maxNumPackets`	Max number of packets	double	Positive integer	1000
`nTimeSamp`*	Channel time samples	double	Positive integer	32
`plotRangeDopplerMap`*	Plot range doppler	double	$\{0, 1\}$	1
`plotVelocity`*	Plot velocity estimation	double	$\{0, 1\}$	1
`plotRange`*	Plot range estimation	double	$\{0, 1\}$	1

*ISAC

Table A.2: Tunable properties of `phy` in `phyConfig.txt`.

Parameter	Description	Data Type	Valid Range	Default
`phyMode`	PHY mode	char — string	'OFDM', 'SC'	'OFDM'
`lenPsduByt`	Length of PSDU.	double	positive scalar	4096
`giType`	Guard Interval Type	char — string	'Short', 'Normal,'Long'	'Long'
`numSTSVec`	Num. spatial-time streams	double	Integer in [1,8]	1
`smTypeNDP`	Spatial Mapping Pilots	char — string	'Hadamard', 'Fourier, 'Custom ', 'Direct'	'Direct'
`smTypeDP`	Spatial Mapping Data	char — string	'Hadamard', 'Fourier, 'Custom ', 'Direct'	'Direct'
`mcs`	Modulation-Coding Index	double	Integer in [0,20-21]	6
`analogBeamforming`**	Analog beamforming	char — string		
`dynamicBeamNumber`**	Dynamic stream allocation	double		
`processFlag`	Transceiver mode	double	Integer in [0,5]	1
`symbOffset`	Symbol sync offset	double	Real in [0,1]	0.75
`softCsiFlag`	Soft CSI	double	$\{0, 1\}$	0 (SC), 1 (OFDM)
`ldpcDecMethod`	LDPC decoding method	char — string	'norm-min-sum', 'bp'	'norm-min-sum'

*ISAC **It requires additional dataset provided upon request

Table A.3: Tunable properties of `sens` in `sensConfig.txt`.

Parameter	Description	Data Type	Valid Range	Default
`pri`	Pulse Repetition Interval	double	positive scalar	0.0005
`dopplerFftLen`	FFT length	double	positive integer	64
`pulsesCpi`	Coherent Pulses	double	positive integer	64
`window`	FFT window	double	positive integer	64

Parameter	Description	Data Type	Valid Range	Default
windowLen	FFT window length	double	positive integer	64
overlap	FFT window overlap	double	Real in [0,1]	0

Table A.4: Tunable properties of chan in channelConfig.txt if chanModel = Rayleigh.

Parameter	Description	Data Type	Valid Range	Default
numTaps	Number of taps	double	positive scalar	10
pdpMethodStr	Power delay profile	char — string	PS, Equ, Exp	Exp
tdlTypeChannel	Interpolation	char — string	Impulse, Sinc	Impulse

Table A.5: Tunable properties of chan in channelConfig.txt if chanModel = NIST.

Parameter	Description	Data Type	Valid Range	Default
environmentFile-NameChannel	Environment	char — string	LR, SC, OAH	LR
totalNumber-OfReflection	Reflection order	double	positive integer	2
tdlTypeChannel	Interpolation	char — string	Impulse, Sinc	Impulse

Table A.6: Tunable properties in paaConfigNodeX.txt if chanModel = NIST.

Parameter	Description	Data Type	Valid Range	Default
numAntenna	Number of antennas	double	positive integer	16
Geometry	Geometry	char — string	UniformLinear Uniform-Rectangular	Uniform-Rectangular
numAntennaVert	Number of vertical antennas	double	positive integer	4

References

[1] H. Assasa *et al.* A Collection of Open-source Tools to Simulate IEEE 802.11ad/ay WLAN Networks in Network Simulator ns-3. [Online]. Available: https://github.com/wigig-tools

[2] *IEEE P802.11 Wireless LANs: Channel Models for IEEE 802.11ay*, IEEE Std. IEEE 802.11-15/1150r9, 2015.

[3] A. Bodi, S. Blandino, N. Varshney, J. Zhang, T. Ropitault, M. Lecci, P. Testolina, J. Wang, C. Lai, and C. Gentile, "A quasi-deterministic (Q-D) channel implementation in MATLAB software," https://github.com/wigig-tools/qd-realization, 2021.